UML&AADL'11

An Implementation of the Behavior Annex in the AADL-toolset OSATE2

Jérôme Hugues *jerome.hugues@isae.fr*
Gilles Lasnier *gilles.lasnier@telecom-paristech.fr*
Laurent Pautet *laurent.pautet@telecom-paristech.fr*
Lutz Wrage *lwrage@sei.cmu.edu*

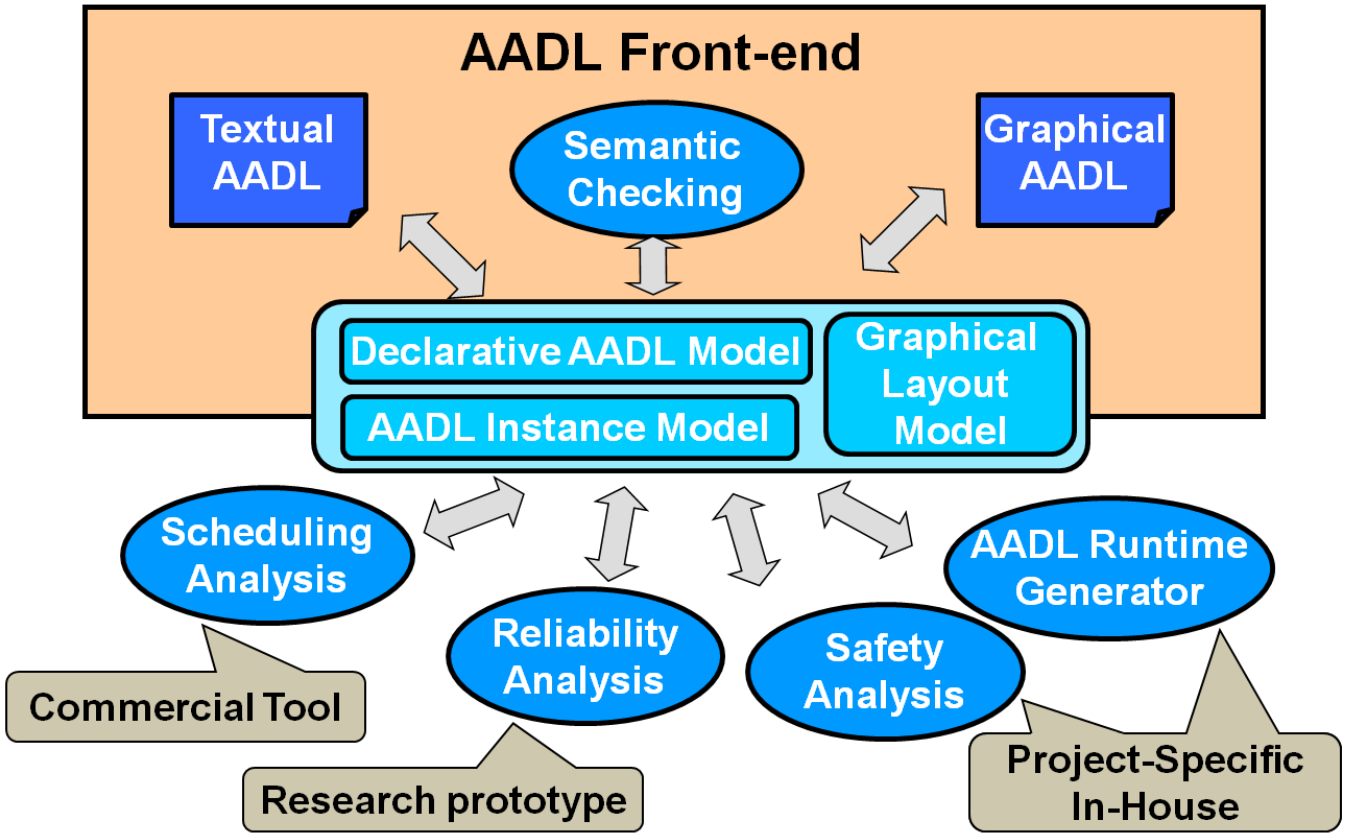
Introduction

- **AADL: *Architecture Analysis & Design Language***
  - Domain Specific Modeling Language (DSML) for High-Integrity (HI) systems
  - Design software + hardware components
  - Each component has a semantic, a textual and a graphical representation
  - AADL extension refine the semantics of an AADL application model
    - Property sets: non-functional properties (thread's period), structured values (strings)
    - Annexes: error modeling annex, behavior annex

- **Several tools to design HI systems with AADL: academic / industrial**
  - **OSATE2** reference toolset to process AADL models (AADL meta-model)
  - Cheddar for schedulability analysis,
  - Ocarina for code generation
  - Topcased (Airbus), Taste (ESA), STOOD (Ellidiss), ADELE…

OSATE2: the AADL reference-toolset

- **OSATE2:** ***The Open-Source AADL Tool Environment***
  - Set of Eclipse-based plug-ins for front-end processing of AADL Models
  - Support the AADLv2

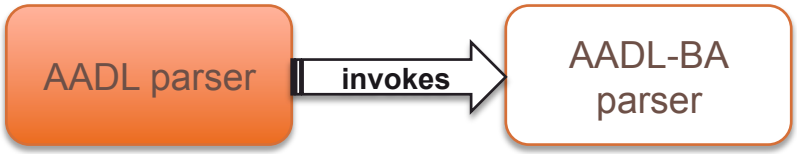


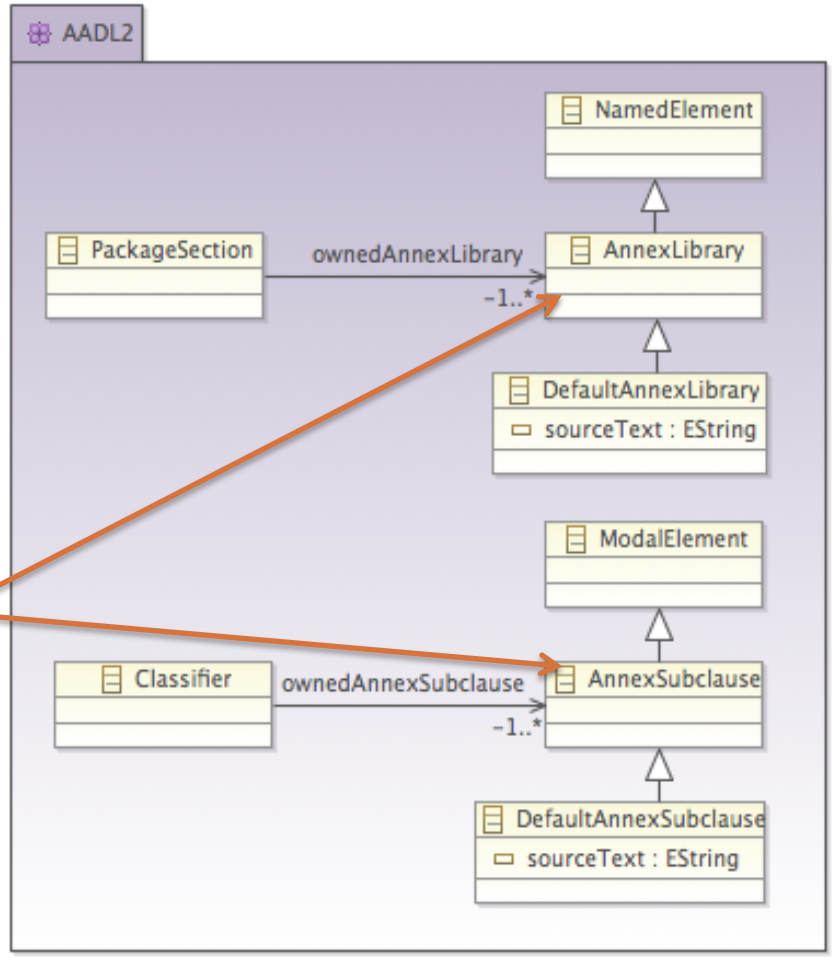
OSATE2: annexes and sub-languages integration

- **Sub-languages in AADL**
  - Used to extend AADL model by integrating complex structures
  - Standardized and published as AADL annex
    - Error modeling annex: error modeling sub-language to define error states and fault propagation
    - Behavior annex: several sub-languages to model detailed component behavior as a state machine

- **Sub-langage support in Osate2**
  - AADL meta-model provides two abstract UML classes extended by the sub-language meta-model
  - Osate2 Annex plug-in implements a registry for sub-language processing: parser, name resolver…

  ⇒ The core-language processing invokes the annex specific functionnalities needed



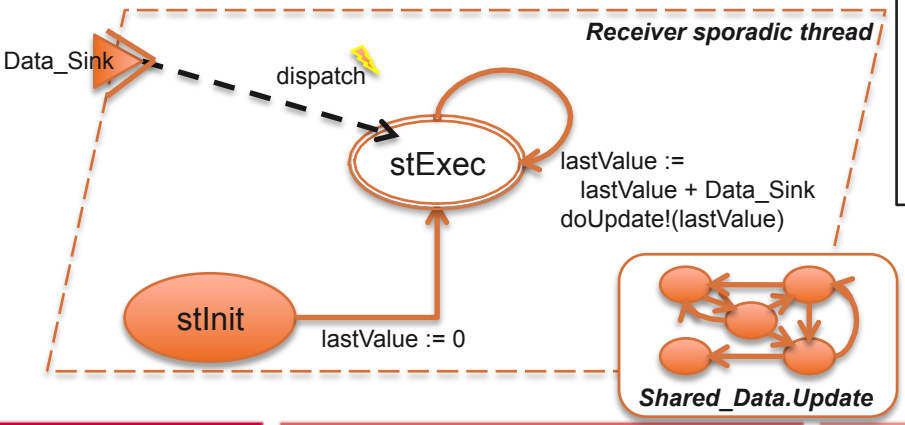


AADL meta-model fragment

AADL Behavior Annex (AADL-BA)

- **Specifies a behavior model for <u>each</u> AADL component**

- **Defines five sub-languages**
  - A ***state/transition automaton*** language
  - A ***thread dispatch*** behavior language
  - A ***component interaction*** language
  - A ***behavior action*** language
  - A ***behavior expression*** specification

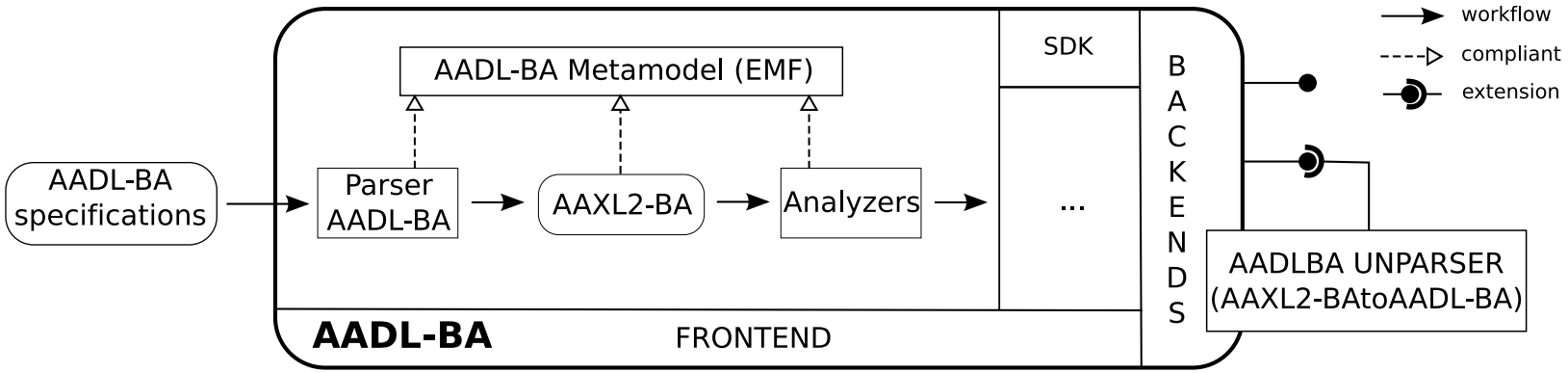



```
thread Receiver
features
 Data_Sink : in event data port dt;
 Shared_Data : requires data access sd;
properties
 Dispatch_Protocol => Sporadic;
end Receiver;

thread implementation Receiver.impl
calls
 toCall: { doUpdate : subprogram Shared_Data.Update; };
annex behavior_specification {**
variables
  lastValue: dt;
states
  stInit: initial state;
  stExec: complete final state;
 transitions
  stInit_Exec: stInit -[ ]-> stExec { lastValue := 0 };
  stExec_Exec: stExec -[on dispatch Data_Sink]-> stExec
        { lastValue := lastValue + Data_Sink;
          doUpdate!(lastValue);  };
 **};
end Receiver.impl;
```

AADL Behavior Annex front-end plug-in

- **Objectives:**
  - Provides an AADL-BA meta-model compliant with the AADL meta-model
  - Verifies syntactic and legality rules defined in the behavior annex standard
  - Verifies relevant semantic rules
  - Verifies the consistency with the AADL application model
  - Builds an AADL-BA tree integrated to the core language tree

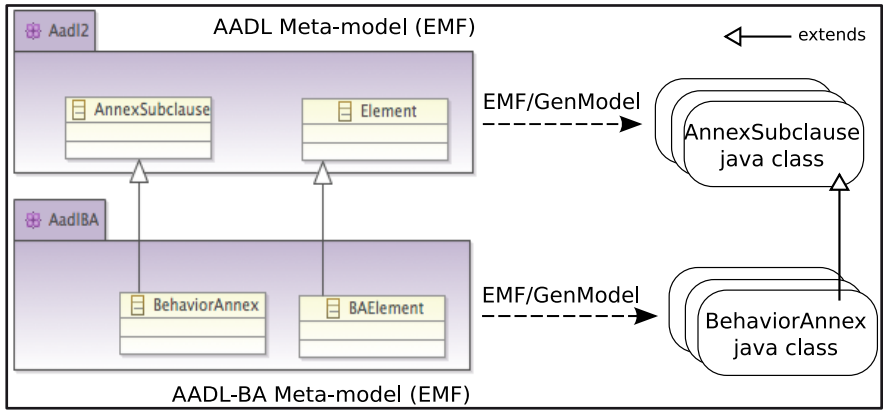
  ⇒ Allows/eases the integration of analysis back-end and analysis tool



AADL-BA plug-in architecture

AADL-BA meta-model specification

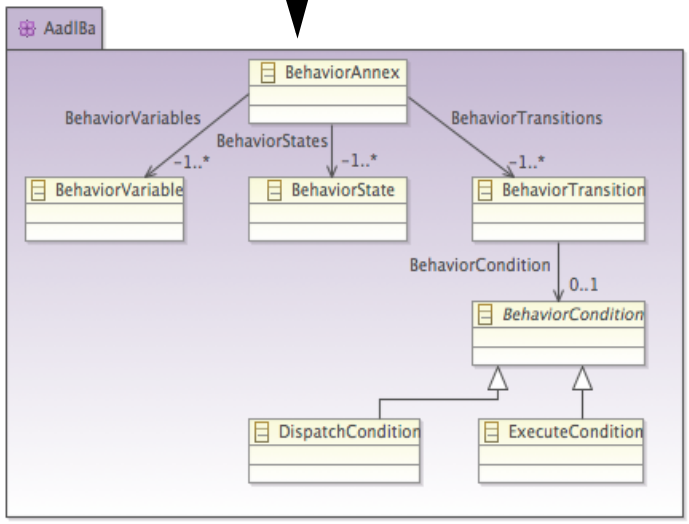
- **A single AADL-BA meta-model specified using EMF**
- **The top-level BehaviorAnnex object extends the abstract AADL AnnexSubclause object**
- **Mapping specification:**
  - R1: Behavior concepts with strong semantics and concrete textual representation (e.g, a BNF) are mapped to meta-model EClass (e.g. a java class)
  - R2: Behavior concepts with weak or no semantic but used to clarify the concept hierarchy are mapped to abstract EClass
  - R3: Behavior concepts belonging to the same family are mapped with respect to the Java's inheritance mechanism.
  - R4: Links to express that a behavior concept requires another behavior concept are mapped to EReferences



AADL-BA meta-model dependencies

```
behavior_annex ::=
   [ variables { behavior_variable }+ ]
   [ states { behavior_state }+ ]
   [ transitions { behavior_transition }+ ]

behavior_transition ::=
   [ transition_identifier [ [ behavior_transition_priority ] ] : ]
     source_state_identifier { , source_state_identifier }*
       -[ behavior_condition ]-> destination_state_identifier
[ { behavior_actions } [ timeout behavior_time ] ] ;

behavior_condition ::= dispatch_condition | execute_condition
```

AADL-BA BNF



AADL-BA meta-model fragment

AADL-BA Parser: BNF -> ANTLR Grammar

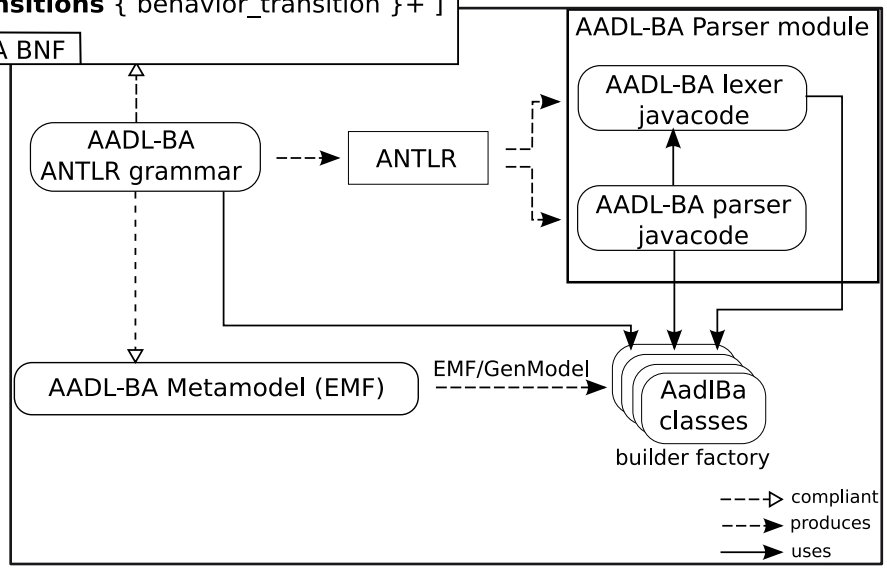
```
behavior_annex returns [BehaviorAnnex ba]
 @init{ ba = AadlBaF.createBehaviorAnnex();
        ba.setLocationReference(new LocationReference(this.getFilename(), input.get(0).getLine())) ;
} :
  ( VARIABLES ( bVars=behavior_variable { ba.getBehaviorVariables().add(bVars);} )+ )?

  ( STATES ( bStates=behavior_state { ba.getBehaviorStates().add(bStates); } )+ )?

  ( TRANSITIONS ( bTrans=behavior_transition {ba.getBehaviorTransitions().add(bTrans); } )+ )?
;
catch [RecognitionException ex] {
  reportError(ex);
  consumeUntil(input,SEMICOLON);
  input.consume();
}
```

AADL-BA BNF mapped to ANTLR grammar rules

```
behavior_annex ::=
   [ variables { behavior_variable }+ ]
   [ states { behavior_state }+ ]
   [ transitions { behavior_transition }+ ]
```

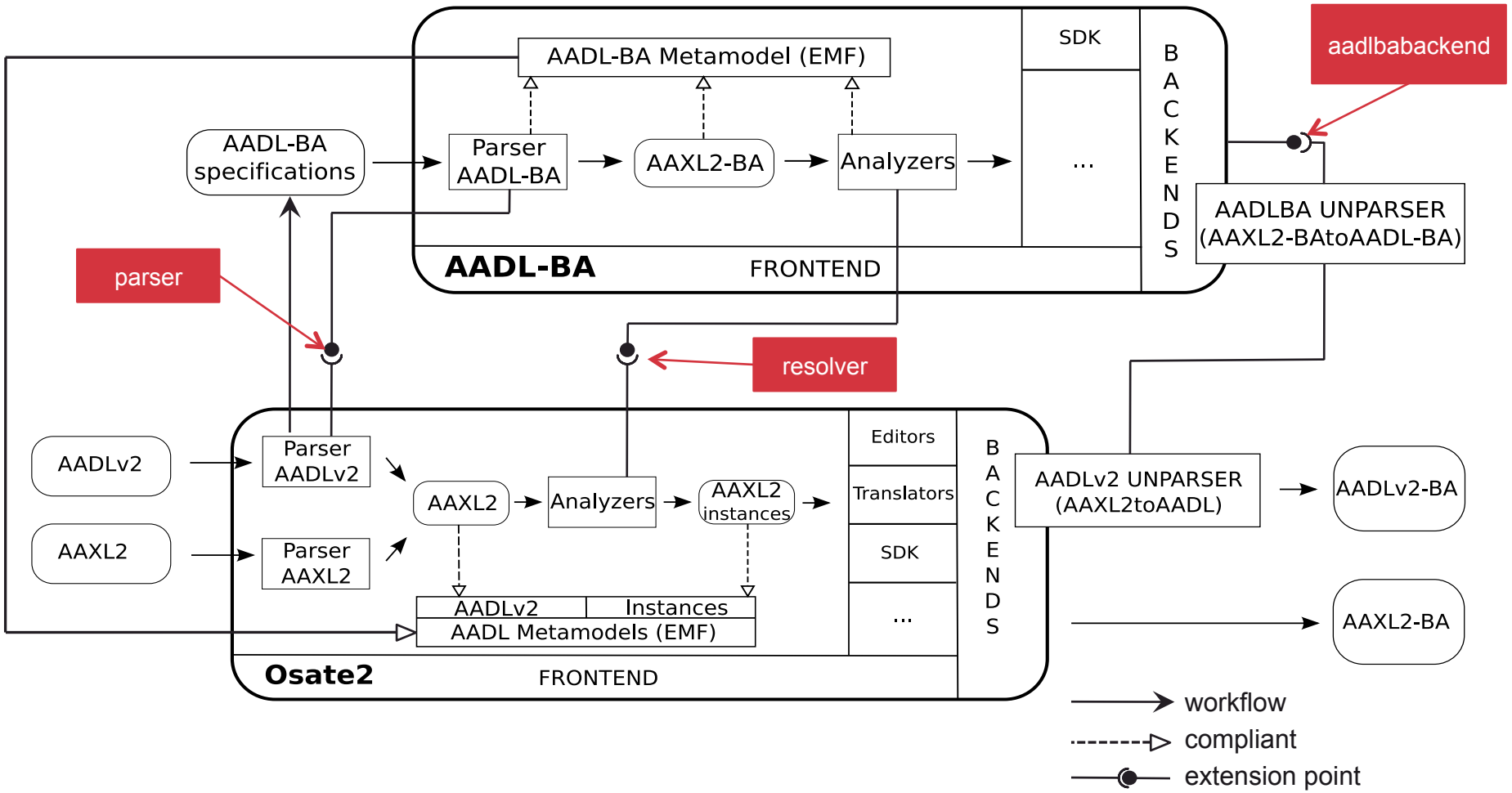
AADL-BA BNF

- **EMF allows to generate the AADL-BA object factory (OF) from the AADL-BA meta-model**
- **ANTLR is used to build lexer and parser:**
  - AADL-BA BNF -> ANTLR Grammar rules
  - OF methods injected in grammar rules
  - ⇒ specify how to build the AADL-BA abstract syntax tree (AST) with AADL-BA objects



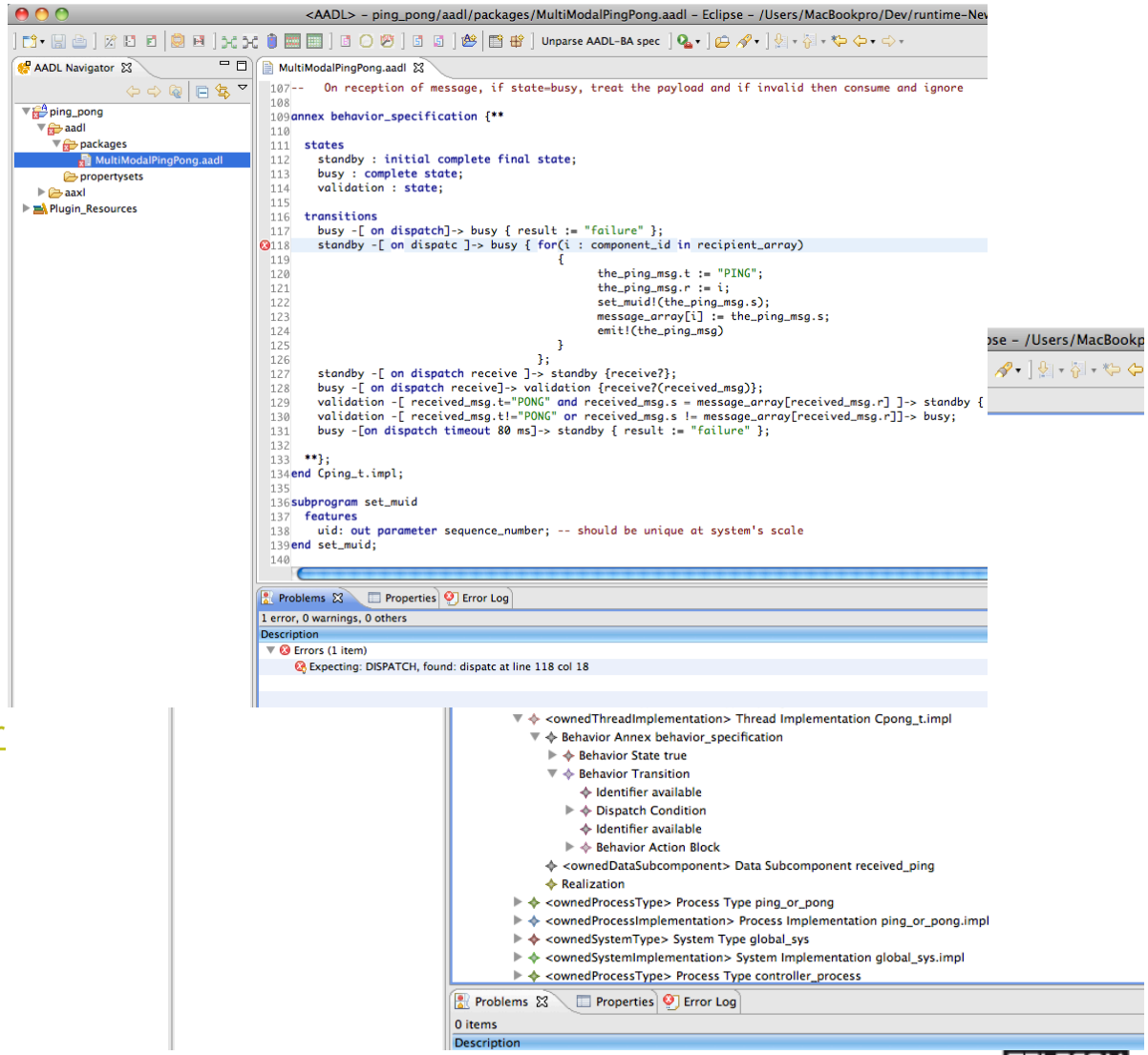
AADL-BA parser module: architecture

AADL-BA plug-in integration in Osate2



AADL-BA plug-in: status

- **Support AADL-BA v0.94 (latest)**
- **Requirements**
  - OSATE2
  - Java 1.6
- **Available at:**
  - AADL-BA plug-in
    http://aadl.telecom-paristech.fr
  - Osate2
    http://www.aadl.info



Conclusion and future works

- **Plug-in operational, support the full language and most of the naming and legality rules**
- **Helped extending OSATE2 to support new annexes in an easy way**
- **Plug-in used to implement a new code generation strategy**
  - Check our paper at ISORC'11
  - Help also as a validation of the meta-model for the AADL-BA itself

- **OSATE2 + AADL-BA released as betas**
  - Full implementation to be released end of 2011